Amendments to the Claims

Claim 1 (currently amended): A method for programmatically enforcing referential integrity constraints among associations between class instances, comprising steps of:

programmatically determining, upon a request to modify an association end for an association between an instance of a first class and an instance of a second class, the association comprising one association end from the instance of the first class to the instance of the second class and another association end from the instance of the second class to the instance of the first class, ~~when evaluating a request to set an association end to reflect an association from an instance of a first class to an instance of a second class,~~ whether the association end to be modified has a single multiplicity or a many multiplicity;

if the association end to be modified has the single multiplicity, programmatically performing the steps of:

disconnecting a previously-existing inverse association end of the association end to be modified, if any;

setting a new inverse association end of the association end to be modified; and

modifying the association end for the association end to be modified,

wherein the steps of disconnecting, setting, and modifying are performed atomically; and

if the association end to be modified has the many multiplicity, programmatically performing the steps of:

modifying the association end to be modified;

disconnecting a previously-existing association end of the association end to be modified, if any; and

setting the inverse association end for the association end to be modified,
wherein the steps of modifying, disconnecting, and setting are performed atomically.
~~setting the requested association end; and~~
~~programmatically modifying an inverse association end of the association to reflect an inverse association from the instance of the second class to the instance of the first class, after disconnecting the inverse association end from an existing instance, if any;~~
~~wherein an ordering of the setting step and the programmatically modifying step depends on an outcome of the determining step.~~

Claims 2 - 4 (canceled)

Claim 5 (currently amended): The method according to Claim 1, further comprising [[steps]] the step of[[:]] serializing the association between the instance of the first class and the instance of the second class by performing steps of:

determining whether the association end to be modified or the inverse association end is a primary end of the association; and

serializing only the primary end of the association during [[a]] the serialization ~~operation~~.

Claim 6 (currently amended): The method according to Claim 1, wherein the method is provided as a single link helper ~~objects~~ object and a multiple link helper object for each association, wherein the single link helper object performs the steps of disconnecting, setting, and modifying for the single multiplicity association end and the multiple link helper object performs the steps of modifying, disconnecting, and setting for the many multiplicity association end.

Claim 7 (currently amended): A computer program product for programmatically enforcing referential integrity constraints among associations between class instances, wherein the computer program product is embodied on one or more computer readable media and comprises computer-readable program code means for:

~~computer-readable program code means for~~ programmatically determining, upon a request to modify an association end for an association between an instance of a first class and an instance of a second class, the association comprising one association end from the instance of the first class to the instance of the second class and another association end from the instance of the second class to the instance of the first class, ~~when evaluating a request to set an association end to reflect an association from an instance of a first class to an instance of a second class,~~ whether the association end to be modified has a single multiplicity or a many multiplicity;

~~computer-readable program code means for setting the requested association end; and~~

if the association end to be modified has the single multiplicity, programmatically performing the steps of:

disconnecting a previously-existing inverse association end of the association end to be modified, if any;

setting a new inverse association end of the association end to be modified; and

modifying the association end for the association end to be modified,

wherein the steps of disconnecting, setting, and modifying are performed atomically; and

if the association end to be modified has the many multiplicity, programmatically performing the steps of:

modifying the association end to be modified;

disconnecting a previously-existing association end of the association end to be modified, if any; and

setting the inverse association end for the association end to be modified,

wherein the steps of modifying, disconnecting, and setting are performed atomically.

~~computer-readable program code means for programmatically modifying an inverse association end of the association to reflect an inverse association from the instance of the second class to the instance of the first class, after disconnecting the inverse association end from an existing instance, if any;~~

~~wherein an ordering of operating the computer-readable program code means for setting and the computer-readable program code means for programmatically modifying depends on an outcome of the computer-readable program code means for determining.~~

Claims 8 - 9 (canceled)

Claim 10 (currently amended): The computer program product according to Claim 7, further comprising computer-readable program code means for serializing the association between the instance of the first class and the instance of the second class by performing steps of:

~~computer-readable program code means for~~ determining whether the association end to be modified or the inverse association end is a primary end of the association; and

computer-readable program code means for serializing only the primary end of the association during [[a]] the serialization~~-operation~~.

Claim 11 (currently amended): A system for programmatically enforcing referential integrity constraints among associations between class instances, comprising means for:

programmatically ~~means for~~ determining, upon a request to modify an association end for an association between an instance of a first class and an instance of a second class, the association comprising one association end from the instance of the first class to the instance of the second class and another association end from the instance of the second class to the instance of the first class, ~~when evaluating a request to set an association end to reflect an association from an instance of a first class to an instance of a second class,~~ whether the association end to be modified has a single multiplicity or a many multiplicity;

if the association end to be modified has the single multiplicity, programmatically performing the steps of:

disconnecting a previously-existing inverse association end of the association end to be modified, if any;

setting a new inverse association end of the association end to be modified; and

modifying the association end for the association end to be modified,

wherein the steps of disconnecting, setting, and modifying are performed atomically; and

if the association end to be modified has the many multiplicity, programmatically performing the steps of:

modifying the association end to be modified;

disconnecting a previously-existing association end of the association end to be modified, if any; and

setting the inverse association end for the association end to be modified,

wherein the steps of modifying, disconnecting, and setting are performed atomically.

~~means for setting the requested association end; and~~

~~means for programmatically modifying an inverse association end of the association to reflect an inverse association from the instance of the second class to the instance of the first class, after disconnecting the inverse association end from an existing instance, if any;~~

~~wherein an ordering of operating the means for setting and the means for programmatically modifying depends on an outcome of the means for determining.~~

Claims 12 - 13 (canceled)

Claim 14 (currently amended): The system according to Claim 11, further comprising means for serializing the association between the instance of the first class and the instance of the second class by performing steps of:

~~means for~~ determining whether the association end to be modified or the inverse association end is a primary end of the association; and

means for serializing only the primary end of the association during [[a]] the serialization ~~operation~~.

Claim 15 (new): The method according to Claim 1, wherein one or more structured markup language representations specify instances of the first class, instances of the second class, and associations between the instances of the first and second classes.

Claim 16 (new): The method according to Claim 15, wherein only one association end for each association between instances is specified in the structured markup language representations.

Claim 17 (new): The method according to Claim 16, wherein the only one association end is an association end designated as a primary end for the association.

Claim 18 (new): The method according to Claim 15, wherein a serialization of results of the request to modify the association end that has the single multiplicity comprises the step of:

determining whether the association end to be modified is a primary end for the association, and if so, programmatically performing the steps of:

removing the representation of the previously-existing inverse association end, if any, from the structured markup representation in which it is specified; and

adding a representation of the new inverse association end.